In the Claims:

Please amend claims 1, 18, 34, 35, and 52 as indicated below.

1. (Currently Amended) A system, comprising:

a processor; and

a memory comprising program instructions, wherein the program instructions are executable by the processor to implement:

a virtual machine;

a default class loader for the virtual machine, configured to:

load classes for code executable within the virtual machine on the system from one or more local locations indicated by a class path of the default class loader;

determine that a class needed to execute the code on the system is not stored in the one or more locations indicated by the class path; and

generate an indication that the class is not loaded;

a remote class loader mechanism configured to:

detect the indication that the class is not loaded;

obtain the class from a remote system via a network; and

store the class in a location indicated by the class path of the default class loader on the system;

wherein the remote class loader mechanism is configured to perform said detect, said obtain, and said store separate from and transparent to the default class loader, and wherein the default class loader is independent from the remote class loader mechanism; and

wherein the default class loader is configured to load the class from the location indicated by the class path, and wherein the default class loader being configured to load the class from the location avoids class conflicts.

2. (Previously presented) The system as recited in claim 1, wherein, to load the class from the location indicated by the class path, the default class loader is configured to:

locate the class stored in the location indicated by the class path; and

load the class from the location for access by the code.

3. (Original) The system as recited in claim 1, wherein the location is a default directory for storing remote classes.

4. (Original) The system as recited in claim 1, wherein the location is a user-specified directory for storing remote classes.

5. (Previously presented) The system as recited in claim 1, wherein said indication is an exception generated by the code and indicating that the class is not stored in the one or more locations indicated by the class path.

6. (Original) The system as recited in claim 1, wherein, to obtain the class from a remote system, the remote class loader mechanism is further configured to send a message requesting the class to one or more remote systems, wherein the message comprises information about the class for identifying a class file on the remote system that comprises the requested class.

7. (Original) The system as recited in claim 1, wherein, to obtain the class from a remote system, the remote class loader mechanism is further configured to:

send a message requesting the class to the remote system; and

receive the class from the remote system in one or more messages in response to the message.

8. (Original) The system as recited in claim 1, wherein, to obtain the class from a remote system, the remote class loader mechanism is further configured to:

broadcast a message requesting the class to one or more remote systems including the remote system on the network; and

receive the class from the remote system in one or more messages in response to the broadcast message.

9. (Previously presented) The system as recited in claim 8, wherein the one or more remote systems and the system are member peers of a peer group in a peer-to-peer network environment.

10. (Canceled)

11. (Previously presented) The system as recited in claim 1, wherein the virtual machine is a Java Virtual Machine (JVM).

12. (Original) The system as recited in claim 1, wherein the code is in a bytecode computer language.

13. (Original) The system as recited in claim 1, wherein the code is Java code.

14. (Original) The system as recited in claim 1, wherein the system and the remote system are peer nodes configured to participate in a peer-to-peer environment on the network.

15. (Original) The system as recited in claim 1, wherein the system and the remote system are configured to participate as peer nodes in a peer-to-peer environment on the network in accordance with one or more peer-to-peer platform protocols for enabling the peer nodes to discover each other, communicate with each other, and cooperate with each other to form peer groups in the peer-to-peer environment.

16. (Original) The system as recited in claim 1, wherein the code is a code fragment of an application configured for execution on the system, and wherein the remote system is a node in a distributed computing framework that comprises the application and is configured to provide computer-executable code fragments of the application to two or more other systems to run the code fragments in parallel to execute the application.

17. (Original) The system as recited in claim 1, wherein the system and the remote system are configured to participate in a distributed computing system on the network for submitting computational tasks in a distributed heterogeneous networked environment that utilizes peer groups to decentralize task dispatching and post-processing functions and enables a plurality of jobs to be managed and run simultaneously.

18. (Currently Amended) A distributed computing system, comprising:

a master node configured to provide computer-executable code fragments of an application to a plurality of worker nodes on a network, wherein the code fragments are configured to run tasks in parallel on two or more of the plurality of worker nodes to perform a job;

a worker node configured to receive a code fragment from the master peer node;

wherein the worker node comprises a virtual machine and a default class loader for the virtual machine, wherein the default class loader is configured to:

load classes for the code fragment executable within the virtual machine from one or more local locations indicated by a class path of the default class loader;

determine that a class needed to execute the code fragment is not stored in the one or more locations indicated by the class path;

wherein the worker node further comprises a remote class loader configured to:

detect that the class is not loaded;

obtain the class from a remote node via the network; and

store the class in a location indicated by the class path of the default class loader on the worker node;

wherein the remote class loader is configured to perform said detect, said obtain, and said store separate from and transparent to the default class loader, and wherein the default class loader is independent from the remote class loader; and

wherein the default class loader is further configured to load the class from the location indicated by the class path, and wherein the default class loader being configured to load the class from the location avoids class conflicts.

19. (Previously presented) The distributed computing system as recited in claim 18, wherein, to load the class from the location indicated by the class path, the default class loader is configured to:

locate the class stored in the location indicated by the class path; and

load the class from the location for access by the code fragment.

20. (Original) The distributed computing system as recited in claim 18, wherein the location is a default directory for storing remote classes.

21. (Original) The distributed computing system as recited in claim 18, wherein the location is a user-specified directory for storing remote classes.

22. (Previously presented) The distributed computing system as recited in claim 18, wherein, to detect that the class is not loaded, the remote class loader is further configured to detect an exception generated by the code fragment and indicating that the class is not on the worker node.

23. (Previously presented) The distributed computing system as recited in claim 18, wherein, to obtain the class from a remote node, the remote class loader is further configured to send a message requesting the class to the remote node, wherein the message comprises information about the class for identifying a class file that comprises the requested class.

24. (Previously presented) The distributed computing system as recited in claim 18, wherein the master node is the remote node, and wherein, to obtain the class from a remote node, the remote class loader is further configured to:

send a message requesting the class to the master node; and

receive the class from the master node in one or more messages in response to the message.

25. (Previously presented) The distributed computing system as recited in claim 18, wherein, to obtain the class from a remote node, the remote class loader is further configured to:

broadcast a message requesting the class to one or more remote nodes on the network; and

receive the class from the remote node in one or more messages in response to the broadcast message.

26. (Original) The distributed computing system as recited in claim 25 wherein the one or more remote nodes, the worker node, and the master nodes are member peers of a peer group in a peer-to-peer network environment.

27. (Original) The distributed computing system as recited in claim 25, wherein the one or more remote nodes are worker nodes configured to receive code fragments from the master node.

28. (Canceled)

29. (Previously presented) The distributed computing system as recited in claim 18, wherein the virtual machine is a Java Virtual Machine (JVM).

30. (Original) The distributed computing system as recited in claim 18, wherein the code fragment is in a bytecode computer language.

31. (Original) The distributed computing system as recited in claim 18, wherein the code fragment is Java code.

32. (Original) The distributed computing system as recited in claim 18, wherein the worker node and the master node are peer nodes configured to participate in a peer-to-peer environment on the network.

33. (Original) The distributed computing system as recited in claim 18, wherein the worker node and the master node are configured to participate as peer nodes in a peer-to-peer environment on the network in accordance with one or more peer-to-peer platform protocols for enabling the peer nodes to discover each other, communicate with each other, and cooperate with each other to form peer groups in the peer-to-peer environment.

34. (Currently Amended) A system, comprising:

a default class loader means for a virtual machine, wherein the default class loader means is configured to load classes for code executable within the virtual machine on the system from one or more local locations indicated by a class path of the default class loader means;

means for determining that a class needed to execute the code on the system is not stored in the one or more locations indicated by the class path;

means for obtaining the class from a remote system via a network; and

means for storing the class in a location on the system indicated by the class path of the default class loader means;

wherein said means for determining, said means for obtaining, and said means for storing are configured to operate separate from and transparent to the default class loader, and wherein the default class loader means is independent from said means for determining, said means for obtaining, and said means for storing; and

wherein the default class loader means is configured to load the class from the location indicated by the class path, and wherein the default class loader means being configured to load the class from the location avoids class conflicts.

35. (Currently Amended) A method, comprising:

loading classes for code executing within a virtual machine on a system from one or more local locations indicated by a class path of a default class loader for the virtual machine;

determining that a class needed to execute the code on the system is not stored in the one or more locations indicated by the class path;

generating an indication that the class is not loaded;

detecting the indication that the class is not loaded;

obtaining the class from a remote system via a network in response to said detecting;

storing the class in a location indicated by the class path of the default class loader on the system;

wherein said detecting, said obtaining, and said storing are performed separate from and transparent to the default class loader, and wherein the default class loader is independent from said detecting, said obtaining, and said storing; and

the default class loader loading the class from the location indicated by the class path, wherein the default class loading the class from the location avoids class conflicts.

36. (Previously presented) The method as recited in claim 35, wherein said loading the class from the location indicated by the class path further comprises:

the default class loader locating the class stored in the location indicated by the class path; and

the default class loader loading the class from the location for access by the code.

37. (Original) The method as recited in claim 35, wherein the location is a default directory for storing remote classes.

38. (Original) The method as recited in claim 35, wherein the location is a user-specified directory for storing remote classes.

39. (Previously presented) The method as recited in claim 35, said indication is an exception generated by the code and indicating that the class is not stored in the one or more locations indicated by the class path.

40. (Original) The method as recited in claim 35, wherein obtaining the class from a remote system comprises sending a message requesting the class to one or more remote systems, wherein the message comprises information about the class for identifying a class file on the remote system that comprises the requested class.

41. (Original) The method as recited in claim 35, wherein said obtaining the class from a remote system comprises:

sending a message requesting the class to the remote system; and

receiving the class from the remote system in one or more messages in response to the message.

42. (Original) The method as recited in claim 35, wherein said obtaining the class from a remote system comprises:

broadcasting a message requesting the class to one or more remote systems on the network; and

receiving the class from the remote system in one or more messages in response to the broadcast message.

43. (Original) The method as recited in claim 42, wherein the one or more remote systems and the system are member peers of a peer group in a peer-to-peer network environment.

44. (Canceled)

45. (Previously presented) The method as recited in claim 35, wherein the virtual machine is a Java Virtual Machine (JVM).

46. (Original) The method as recited in claim 35, wherein the code is in a bytecode computer language.

47. (Original) The method as recited in claim 35, wherein the code is Java code.

48. (Original) The method as recited in claim 35, wherein the system and the remote system are peer nodes configured to participate in a peer-to-peer environment on the network.

49. (Original) The method as recited in claim 35, wherein the system and the remote system are configured to participate as peer nodes in a peer-to-peer environment on the network in accordance with one or more peer-to-peer platform protocols for enabling the peer nodes to discover each other, communicate with each other, and cooperate with each other to form peer groups in the peer-to-peer environment.

50. (Original) The method as recited in claim 35, wherein the code is a code fragment of an application configured for execution on the system, and wherein the remote system is a node in a distributed computing framework that comprises the application and is configured to provide computer-executable code fragments of the application to two or more other systems to run the code fragments in parallel to execute the application.

51. (Original) The method as recited in claim 35, wherein the system and the remote system are configured to participate in a distributed computing system on the network for submitting computational tasks in a distributed heterogeneous networked environment that utilizes peer groups to decentralize task dispatching and post-processing functions and enables a plurality of jobs to be managed and run simultaneously.

52. (Currently Amended) A computer-accessible storage medium, comprising program instructions, wherein the program instructions are computer-executable to implement:

loading classes for code executing within a virtual machine on a system from one or more local locations indicated by a class path of a default class loader for the virtual machine;

determining that a class ~~is~~ needed to execute the code on the system is not stored in the one or more locations indicated by the class path;

generating an indication that the class is not loaded;

detecting the indication that the class is not loaded;

obtaining the class from a remote system via a network;

storing the class in a location indicated by the class path of the default class loader on the system;

wherein said detecting, said obtaining, and said storing are performed separate from and transparent to the default class loader, and wherein the default class loader is independent from said detecting, said obtaining, and said storing; and

the default class loader loading the class from the location indicated by the class path, and wherein the default class loader loading the class from the location avoids class conflicts.

53. (Previously presented) The computer-accessible storage medium as recited in claim 52, wherein, in said loading the class from the location indicated by the class path, the program instructions are further computer-executable to implement:

the default class loader locating the class stored in the location indicated by the class path; and

the default class loader loading the class from the location for access by the code.

54. (Previously presented) The computer-accessible storage medium as recited in claim 52, wherein the location is a default directory for storing remote classes.

55. (Previously presented) The computer-accessible storage medium as recited in claim 52, wherein the location is a user-specified directory for storing remote classes.

56. (Previously presented) The computer-accessible storage medium as recited in claim 52, wherein said indication is an exception generated by the code and indicating that the class is not stored in the one or more locations indicated by the class path..

57. (Previously presented) The computer-accessible storage medium as recited in claim 52, wherein, in said obtaining the class from a remote system, the program instructions are further computer-executable to implement sending a message requesting the class to one or more remote systems, wherein the message comprises information about the class for identifying a class file on the remote system that comprises the requested class.

58. (Previously presented) The computer-accessible storage medium as recited in claim 52, wherein, in said obtaining the class from a remote system, the program instructions are further computer-executable to implement:

sending a message requesting the class to the remote system; and

receiving the class from the remote system in one or more messages in response to the message.

59. (Previously presented) The computer-accessible storage medium as recited in claim 52, wherein, in said obtaining the class from a remote system, the program instructions are further computer-executable to implement:

broadcasting a message requesting the class to one or more remote systems on the network; and

receiving the class from the remote system in one or more messages in response to the broadcast message.

60. (Previously presented) The computer-accessible storage medium as recited in claim 59, wherein the one or more remote systems and the system are member peers of a peer group in a peer-to-peer network environment.

61. (Canceled)

62. (Previously presented) The computer-accessible storage medium as recited in claim 52, wherein the virtual machine is a Java Virtual Machine (JVM).

63. (Previously presented) The computer-accessible storage medium as recited in claim 52, wherein the code is in a bytecode computer language.

64. (Previously presented) The computer-accessible storage medium as recited in claim 52, wherein the code is Java code.

65. (Previously presented) The computer-accessible storage medium as recited in claim 52, wherein the system and the remote system are peer nodes configured to participate in a peer-to-peer environment on the network.

66. (Previously presented) The computer-accessible storage medium as recited in claim 52, wherein the system and the remote system are configured to participate as peer

nodes in a peer-to-peer environment on the network in accordance with one or more peer-to-peer platform protocols for enabling the peer nodes to discover each other, communicate with each other, and cooperate with each other to form peer groups in the peer-to-peer environment.

67. (Previously presented) The computer-accessible storage medium as recited in claim 52, wherein the code is a code fragment of an application configured for execution on the system, and wherein the remote system is a node in a distributed computing framework that comprises the application and is configured to provide computer-executable code fragments of the application to two or more other systems to run the code fragments in parallel to execute the application.

68. (Previously presented) The computer-accessible storage medium as recited in claim 52, wherein the system and the remote system are configured to participate in a distributed computing system on the network for submitting computational tasks in a distributed heterogeneous networked environment that utilizes peer groups to decentralize task dispatching and post-processing functions and enables a plurality of jobs to be managed and run simultaneously.